(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **11338736 A**

(43) Date of publication of application: **10.12.99**

(51) Int. Cl **G06F 11/28**

(21) Application number: **10305302**

(22) Date of filing: **27.10.98**

(30) Priority: **22.12.97 US 97 995629**

(71) Applicant: **INTERNATL BUSINESS MACH CORP <IBM>**

(72) Inventor: **JAMES LYNN TAYLER**

(54) **METHOD AND COMPUTER SYSTEM FOR VERIFYING GRAPHICAL DISPLAY OUTPUT OF APPLICATION**

(57) Abstract:

PROBLEM TO BE SOLVED: To verify that the graphical output of a given software application is consistent over all operating system platforms by comparing pieces of given information generated for the operating systems respectively.

SOLUTION: When the software application is executed on the respective operating systems, one group of graphical display outputs is captured at a given execution point. Each pair of graphical output frames are processed for the respective operating systems to generate the difference or a delta value. The specific delta value includes information masking the constitution of the graphical display outputs to the respective operating systems which do not change over the couple of frames when the application is executed at the given execution point.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－338736

(43)公開日　平成11年(1999)12月10日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　11/28	３４０	Ｇ０６Ｆ　11/28	３４０Ａ

審査請求　有　請求項の数33　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－305302

(22)出願日　平成10年(1998)10月27日

(31)優先権主張番号　０８／９９５６２９

(32)優先日　1997年12月22日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　ジェイムズ・リン・テイラー
アメリカ合衆国76530、テキサス州、グランガー、カウンティ・ロード・156　1975

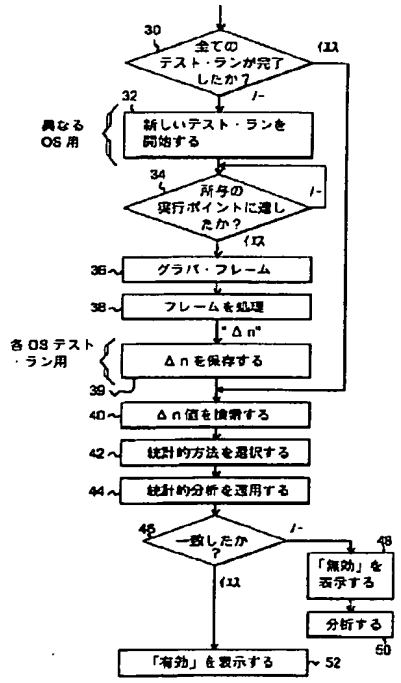
(74)代理人　弁理士　坂口　博　（外１名）

(54)【発明の名称】　アプリケーションのグラフィカル・ディスプレイ出力を検証する方法及びコンピュータ・システム

(57)【要約】　（修正有）

【課題】アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する。

【解決手段】アプリケーションにおける所与の実行ポイントにおいて連続的なグラフィカル・ディスプレイ出力フレームを捕捉し、それらのフレームの対全体にわたって時不変性であるそれぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクするデルタ値を発生するように、そのフレームの対が処理される。各テスト・ラン中に発生されたデルタ値は、所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証するために、統計的分析を使用して比較される。



1

【特許請求の範囲】

【請求項１】所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する方法にして、

前記それぞれのオペレーティング・システムの各々に基づいて前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて一対のグラフィカル・ディスプレイ出力フレームを捕捉するステップと、

前記それぞれのオペレーティング・システムの各々に対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために前記グラフィカル・ディスプレイ出力フレームの各対を処理するステップと、

所与の前記ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって所与の基準を満たすことを検証するために、それぞれのオペレーティング・システムの各々に対して発生された所与の情報を比較するステップと、

を含む方法。

【請求項２】所与の前記ソフトウエア・アプリケーションはＪａｖａで書かれたアプリケーションであることを特徴とする請求項１に記載の方法。

【請求項３】前記一対のフレームは連続的なグラフィカル・ディスプレイ出力フレームであることを特徴とする請求項１に記載の方法。

【請求項４】前記比較するステップは統計的アルゴリズムの実行を含むことを特徴とする請求項１に記載の方法。

【請求項５】前記所与の基準は前記所与の情報が所与の信頼性レベル内の情報と等しいかどうかをテストすることを特徴とする請求項１に記載の方法。

【請求項６】前記所与の信頼性レベルはユーザ選択可能であることを特徴とする請求項５に記載の方法。

【請求項７】前記処理するステップは前記一対のフレームを含むフレームにおける排他的ＯＲ論理オペレーションの遂行を含むことを特徴とする請求項１に記載の方法。

【請求項８】前記排他的ＯＲ論理オペレーションはビット単位の基準で遂行されることを特徴とする請求項７に記載の方法。

【請求項９】Ｊａｖａアプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する方法にして、

前記それぞれのオペレーティング・システムの各々に基づいて、Ｊａｖａアプリケーションにおける所与の実行ポイントにおいて連続的なグラフィカル・ディスプレイ出力フレームを捕捉するステップと、

2

前記それぞれのオペレーティング・システムの各々に対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクするデルタ値を発生するために、前記連続的なグラフィカル・ディスプレイ出力フレームを処理するステップと、

前記Ｊａｖａアプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するために、前記それぞれのオペレーティング・システムの各々に対して発生された前記デルタ値を統計的に比較するステップと、

を含む方法。

【請求項１０】前記ＪａｖａアプリケーションはＪａｖａアプレットであることを特徴とする請求項９に記載の方法。

【請求項１１】前記処理するステップはビット単位の排他的ＯＲ論理オペレーションであることを特徴とする請求項９に記載の方法。

【請求項１２】前記比較するステップは前記デルタ値に関して統計的分析を行うことを含むことを特徴とする請求項９に記載の方法。

【請求項１３】前記統計的分析はパターン・マッチング・アルゴリズムであることを特徴とする請求項１２記載の方法。

【請求項１４】前記統計的分析はファジー・ロジック・アルゴリズムであることを特徴とする請求項１２記載の方法。

【請求項１５】前記統計的分析は神経ネットワーク・アルゴリズムであることを特徴とする請求項１２記載の方法。

【請求項１６】所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する場合に使用するためのコンピュータ読み取り可能な媒体におけるコンピュータ・プログラム製品にして、

前記それぞれのオペレーティング・システムの各々に基づいて、前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて動作し、グラフィカル・ディスプレイ出力フレームの対を捕捉するための手段と、

前記それぞれのオペレーティング・システムに対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために、前記捕捉手段に応答して前記グラフィカル・ディスプレイ出力フレームの各対を処理するための手段と、

前記所与のソフトウエア・アプリケーションのグラフィ

カル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するために、前記処理するための手段に応答して、前記それぞれのオペレーティング・システムの各々に対して発生された所与の情報を比較するための手段と、
を含むコンピュータ・プログラム製品。
【請求項１７】前記所与のソフトウエア・アプリケーションはＪａｖａで書かれたアプリケーションであることを特徴とする請求項１６に記載のコンピュータ・プログラム製品。
【請求項１８】前記捕捉するために手段はビデオ・フレーム・グラバであることを特徴とする請求項１６に記載のコンピュータ・プログラム製品。
【請求項１９】前記処理するための手段は排他的ＯＲ論理オペレーションであることを特徴とする請求項１６に記載のコンピュータ・プログラム製品。
【請求項２０】前記比較するための手段は統計的分析であることを特徴とする請求項１７に記載のコンピュータ・プログラム製品。
【請求項２１】前記統計的分析はパターン・マッチング・アルゴリズムであることを特徴とする請求項２０に記載のコンピュータ・プログラム製品。
【請求項２２】前記統計的分析はファジー・ロジック・アルゴリズムであることを特徴とする請求項２０に記載のコンピュータ・プログラム製品。
【請求項２３】前記統計的分析は神経ネットワーク・アルゴリズムであることを特徴とする請求項２０に記載のコンピュータ・プログラム製品。
【請求項２４】プロセッサと、
所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システムに全体にわたって整合していることを検証する場合に使用するためのツールと、
を含み、
前記ツールは、
前記それぞれのオペレーティング・システムの各々に基づいて、前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて動作し、一対のグラフィカル・ディスプレイ出力フレームを捕捉するための手段と、
前記それぞれのオペレーティング・システムに対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために、前記捕捉手段に応答して前記グラフィカル・ディスプレイ出力フレームの各対を処理するための手段と、
前記所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証

するように、前記処理するための手段に応答して、前記それぞれのオペレーティング・システムの各々に対して発生された所与の情報を比較するための手段と、
を含むコンピュータ。
【請求項２５】前記所与のソフトウエア・アプリケーションはＪａｖａで書かれたアプリケーションであることを特徴とする請求項２４に記載のコンピュータ。
【請求項２６】前記捕捉するために手段はビデオ・フレーム・グラバであることを特徴とする請求項２４に記載のコンピュータ。
【請求項２７】前記処理するための手段は排他的ＯＲ論理オペレーションであることを特徴とする請求項２４に記載のコンピュータ。
【請求項２８】前記比較するための手段は統計的分析であることを特徴とする請求項２４に記載のコンピュータ。
【請求項２９】オブジェクト指向プラットフォーム独立のプログラミング環境において最初に書かれたソフトウエア・アプリケーションが第１及び第２オペレーティング・システムのそれぞれにおいて稼動するそれぞれのウインドウ環境において実行される時、前記ソフトウエア・アプリケーションが整合しているグラフィカル出力を有することを検証する方法にして、
前記それぞれのオペレーティング・システムの各々に基づいて、前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて一対のグラフィカル・ディスプレイ出力フレームを捕捉するステップと、
前記それぞれのオペレーティング・システムに対して、前記一対のグラフィカル・ディスプレイ出力フレーム全体にわたって時不変性であるそれぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために前記グラフィカル・ディスプレイ出力フレームの各対を処理するステップと、
前記所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するために、前記それぞれのオペレーティング・システムの各々に対して発生された所与の情報を統計的に比較するステップと、
を含む方法。
【請求項３０】前記オブジェクト指向プラットフォーム独立のプログラミング環境はＪａｖａであることを特徴とする請求項２９に記載の方法。
【請求項３１】前記ソフトウエア・アプリケーションがそれぞれのテスト・システム・コンフィギュレーション全体にわたって実行される時、前記ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が整合していることを検証する方法にして、
前記ソフトウエア・アプリケーションがそれぞれのテス

ト・システム構成の各々に基づいて実行される時、前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて、連続的なグラフィカル・ディスプレイ出力フレームを捕捉するステップと、

それぞれの前記テスト・システム・コンフィギュレーションに対して、前記グラフィカル・ディスプレイ出力フレームの対にわたって時不変性であるグラフィカル・ディスプレイ出力の構成をマスクするデルタ値を発生するために前記連続的なグラフィカル・ディスプレイ出力フレームを処理するステップと、

前記ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記テスト・システム・コンフィギュレーション全体にわたって整合していることを検証するために、前記テスト・システム・コンフィギュレーションの各々に対して発生された前記デルタ値を統計的に比較するステップと、

を含む方法。

【請求項３２】前記テスト・システム・コンフィギュレーションの少なくとも２つは所与のオペレーティング・システムを利用することを特徴とする請求項３１に記載の方法。

【請求項３３】前記テスト・システム・コンフィギュレーションの少なくとも２つは相異なるオペレーティング・システムを利用することを特徴とする請求項３１に記載の方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般的に云えば、コンピュータ・ソフトウエア開発ツールに関するものであり、更に詳しく云えば、特定のソフトウエア・アプリケーションからのグラフィカル出力が複数の相異なるタイプのオペレーティング・システム・プラットフォーム全体にわたって「整合している」ことを検証するための方法に関するものである。

【０００２】

【従来の技術】米国のサン・マイクロシステムズ社によって最初に開発されたＪａｖａは、ソフトウエア・プログラムを開発、試験、及び保守するために使用されるオブジェクト指向のマルチスレッドの移植可能なプラットフォーム独立の保護プログラミング環境である。Ｊａｖａプログラムは、インターネットのマルチメディア情報検索システムであるワールド・ワイド・ウェブにおける拡張使用を提供した。これらのプログラムは、アプレットとして知られた、Ｊａｖａイネーブルのウェブ・ブラウザ又はアプレット・ビューアにおいて稼動するフル・フィーチャの対話式スタンドアロン・アプリケーション及び小型のプログラムを含む。

【０００３】多くの企業がＪａｖａアプリケーション又はアプレットを書き始めている。しかし、これらの企業は、種々のオペレーティング・システムを持ったコンピュータ・システムにおいて大きな投資も行っている。従って、例えば、或る特定の企業のネットワークは、ＩＢＭ　ＯＳ／２（商標）、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ９５、Ｍｉｃｒｏｓｏｆｔ　ＷｉｎｄｏｗｓＮＴ　４．０、Ｕｎｉｘ、ＡＩＸ（商標）、ＯＳ／４００等のような種々のオペレーティング・システムを稼動させる多くの機械をサポートすることがある。そのようなＪａｖａベースのプログラムは、ウインドウ、メニュー、ビットマップ、アイコン、及びグラフィカル・ユーザ・インターフェース（ＧＵＩ）を構成する他の基本的な制御エレメントのような汎用インターフェース表示エレメント並びにプログラム特有の表示エレメントを含むグラフィカル・ディスプレイ出力を発生する。全体的に或いは部分的に、それらのエレメントはアプリケーションのための「グラフィカル出力」を定義し、アプリケーションのグラフィカル出力が種々のオペレーティング・システム・プラットフォーム全体にわたって（及び他のハードウエア／ソフトウエアの相違を持った同種のプラットフォーム全体にわたって）整合していることがソフトウエア設計者の設計目的である。

【０００４】現在のところ、この分野は、特定のＪａｖａベースのアプリケーション又はアプレットのオペレーションが多くのオペレーティング・システム・プラットフォーム上では同じ（又は、十分に同じ）であるということを検証する効率的な手段又は方法を提供していない。同様に、この分野は、Ｊａｖａアプリケーション又はアプレットの多重実行の出力が所与の信頼性レベルにおける「同等」であることを検証するための如何なる適当な技法も提供していない。そのような技法なしに、新しいＪａｖａベースのアプリケーション及びアプレットの試験を自動化することは困難であった。

【０００５】本発明はこの問題に対処するものである。

【０００６】

【発明が解決しようとする課題】従って、本発明の主たる目的は、所与のソフトウエア・アプリケーションのグラフィカル出力が複数のオペレーティング・システム・プラットフォーム全体にわたって整合していることを検証することにある。

【０００７】本発明の更なる主要な目的は、所与のソフトウエア・アプリケーションのグラフィカル出力が特定のコンピュータ・コンフィギュレーションにおいてアプリケーションの多重実行全体にわたって整合していることを検証することにある。

【０００８】本発明の更にもう１つの主要な目的は、複数のオペレーティング・システム・プラットフォームにおけるＪａｖａベースのアプリケーション又はアプレットが、それによって企業のコンピュータ・ネットワークにおける新しいＪａｖａベースのアプリケーションの設計、開発、試験、及び導入を容易にすることを検証することにある。

【０００９】本発明の更にもう１つの目的は、新しいソフトウエア・アプリケーションを開発する場合に有用な試験方法の自動化を促進することにある。

【００１０】本発明の更にもう１つの目的は、アプリケーションのグラフィカル・ディスプレイ出力を検証するための自動化されたツールを提供することによってＪａｖａプログラミング開発環境を向上させることにある。

【００１１】本発明の重要な目的は、Ｊａｖａプラットフォーム独立のコードの試験及び検証を可能にすることにある。

【００１２】本発明の副産物又は利点は、プラットフォーム独立のプログラミング環境に従って書かれたアプリケーションのようなソフトウエア・アプリケーションを設計、開発、及び試験するために必要な時間及び労力を少なくすることである。

【００１３】本発明の更にもう１つの一般的な目的は、ソフトウエア・アプリケーションの作成と関連した開発時間及び費用を少なくすることにある。

【００１４】

【課題を解決するための手段】本発明のこれらの目的及び他の目的は、所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する方法において達成される。本発明の方法の望ましい実施例によれば、それぞれのオペレーティング・システムの各々に基づいてソフトウエア・アプリケーションが実行される時、一組のグラフィカル・ディスプレイ出力（例えば、一対の隣接するフレーム）がそのソフトウエア・アプリケーションにおける所与の実行ポイントにおいて「捕捉」される。このためには、ビデオ・フレーム・グラバ（ｇｒａｂｂｅｒ）が有用である。そこで、それぞれのオペレーティング・システムの各々に対して、グラフィカル出力フレームの各対が差又は「デルタ」値を発生するために処理される。その特定のデルタ値は、アプリケーションが所与の実行ポイントにおいて実行される時、その一対のフレーム全体にわたって変わらない（即ち、時不変性のである）それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする情報を含んでいる。そのような構成は、例えば、ウインドウ境界線、タイトル・バー、スクロールバー、及び他のそのようなディスプレイ制御エレメントを含み得る。

【００１５】この方法によれば、処理ステップは、フレームに関して、ビット単位対ビット単位に基づいて論理オペレーション（例えば、排他的ＯＲ）を遂行することを含み得る。次に、アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって一様であるか又は「整合している」ことを検証するために、それぞれのオペレーティング・システムに対するデルタ値が比較される。所与のディスプレイ出力は、その出力が所与の信頼性レベル（例えば、９５％の類似性）と同じか或いはそれの範囲内のものに一致する場合、「整合している」と云える。デルタ値の比較は、パターン・マッチング・アルゴリズムのような統計的方法、ファジー・ロジック・アルゴリズムを実現する神経ネットワーク、或いは、他の既知の又は今後開発される統計的方法を使用して行われることが望ましい。別の方法として、１つ又は複数の決定的な方法をこの目的のために使用することもあり得る。

【００１６】本発明の方法は、任意のタイプのソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力を検証するために使用されるが、特定の好適な実施例は、Ｊａｖａベースのアプリケーション又はアプレットの試験及び検証を伴うものである。

【００１７】上記の方法は、特定のアプリケーションのオペレーションが種々のオペレーティング・システム・プラットフォーム全体にわたって同じ（又は、実質的に同じ）であることを検証する。本発明の技法は、所与のアプリケーション（例えば，Ｊａｖａアプリケーション）の多重実行の出力が同種のコンピュータ・システム（例えば、単一のタイプのオペレーティング・システムを有するコンピュータ・システム）全体にわたって同じであることを検証するためにも適用可能である。本発明の方法は、連続的なグラフィカル・ディスプレイ出力フレームが「テスト・ラン」というアプリケーションの間に特定の「実行」ポイントにおいて捕捉されるという点で前述の方法と全く同じであることが望ましい。それぞれのテスト・ランに対して、その連続的なグラフィカル・ディスプレイ出力フレームは、デルタ値を発生するために処理される。前述のように、デルタ値はフレームの対全体にわたって時不変性であるグラフィカル・ディスプレイ出力構成をマスク又は「フィルタ」する。ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が複数のテスト・ランのセット全体にわたって整合していることを検証するために、統計的な方法を使用してそのテスト・ラン（又は、所与のテスト・ランのセット）に対するデルタ値が比較される。

【００１８】以上では、本発明のより関連の深い目的及び特徴の幾つかを概説した。これらの目的及び特徴は、単に、本発明のより顕著な特徴及びアプリケーションのうちの幾つかを示したものに過ぎないと解釈されるべきである。開示された発明を別の方法で適用すること及び本発明を後述のように修正することによって、他の多くの有益な結果を得ることも可能である。従って、以下の好適な実施例の詳細な説明を参照することによって、本発明の他の目的及びより十分な理解が得られるであろう。

【００１９】

【発明の実施の形態】後述するように、本発明の「検証」ツールを構成する１つ又は複数のプロセスは、コン

ピュータ上で又はコンピュータ・ネットワークを介して接続された１つ又は複数のコンピュータ上で実行可能である。図１を参照すると、本発明を実施する場合に使用するためのコンピュータが示される。コンピュータ１０は、プロセッサ１２、オペレーティング・システム１４、オペレーティング・システム・アプリケーション・プログラミング・インターフェース（ＡＰＩ）１６、グラフィカル・ユーザ・インターフェース（ＧＵＩ）１８、及びＲＡＭ２２を有する。本発明の実施例では、そのツールは、Ｊａｖａベースのアプリケーション又はアプレットを検証するために使用され、従って、そのコンピュータは、Ｊａｖａ仮想計算機（ＪＶＭ）インタープリタ２０も任意選択的に含む。ＪＶＭは、命令セットを含み、しかも種々のメモリ・エリアを使用する抽象計算機である。ＪＶＭは、Ｎｅｔｓｃａｐｅ Ｎａｖｉｇａｔｏｒ（商標）及びＭｉｃｒｏｓｏｆｔ Ｉｎｔｅｒｎｅｔ Ｅｘｐｌｏｒｅｒ（商標）の適切なバージョンのような多くの「オフ・ザ・シェルフ」インターネット・ブラウザ・アプリケーションにおいて得られる。ＪＶＭに関する更なる詳細は、Addison Wesley 社発行（１９９７年）の Tim Lindholm 及び Frank Yellin 著「Ｊａｖａ仮想計算機仕様（The Java Virtual Machine Specification）」において得られる。

【００２０】従って、例えば、本発明において使用されるコンピュータは、Ｉｎｔｅｌ（商標）ベース、ＰｏｗｅｒＰＣ（商標）ベース、又はＲＩＳＣベースであって、ＩＢＭ ＯＳ／２（商標）、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ ９５、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ ＮＴ ４.０、Ｕｎｉｘ、ＡＩＸ（商標）、ＯＳ／４００等のようなオペレーティング・システムを含む任意のパーソナル・コンピュータ、或いは、ワークステーション・クライアント又はサーバ・プラットフォームである。代表的なコンピュータは、Ｉｎｔｅｌ ｘ８６ プロセッサ、ＯＳ／２ Ｗａｒｐ バージョン４.０ オペレーティング・システム、及びＪＶＭバージョン１.０.２以上を稼動させる。別の方法として、コンピュータは、ｘ８６ベースのプロセッサ、Ｗｉｎｄｏｗｓ ９５（又は、Ｗｉｎｄｏｗｓ ＮＴ）オペレーティング・システム、及びＪＶＭバージョン１.０.２以上を稼動させる。更に別の方法は、ＰｏｗｅｒＰＣ又はＲＩＳＣベースのプロセッサ，ＡＩＸオペレーティング・システム、及びＪＶＭ１.０.２以上を有するコンピュータを稼動させることである。

【００２１】検証ツール２５はソフトウエアで実現され、特に、Ｊａｖａ仮想計算機インタープリタ２０による実行のためにＪａｖａで書かれることが望ましい。検証ツール２５はＪａｖａベースであることが望ましいけれども、これは本発明の必要条件ではない。しかし、Ｊａｖａベースの検証ツールは、複数の開発プラットフォーム全体にわたって移植可能である。従って、検証ツール・コード自体は、上記のシステムの各々において、たとえそれらが少なくともオペレーティング・システムのレベルでは異種であっても動作可能である。その検証ツールは、それと関連し且つコンピュータによって制御されるテスト・サブシステムを持つことが望ましい。テスト・サブシステム２６は、一般に、コンピュータ・ユーザ・インターフェース及び任意の必要なインターフェース上でビデオ・フレーム出力を捕捉するために使用するための１つ又は複数のビデオ・フレーム・グラバ２８のような適切なハードウエア、制御及び同期化回路及びコンポーネントを含む。ビデオ・グラバはその分野ではよく知られている。テスト・サブシステム２６のコンポーネント又は機能の幾つか又はすべてが、全体的に又は部分的にソフトウエアで、又はコンピュータ自身の内部資源及び周辺装置の使用を通して実現可能である。

【００２２】検証ツール２５は、一般に、開発中（或いは、テスト中）の所与のアプリケーションのオペレーションが複数のオペレーティング・システム・プラットフォーム全体にわたって整合したディスプレイ出力を有するかどうかを検証するように機能する。別の方法では、その検証ツールは、アプリケーションのグラフィカル・ディスプレイ出力が同じオペレーティング・システムを持ったシステムにおける多重実行に整合し或いはそれら全体にわたるものであるかどうかを検証するために有用である。その分野ではよく知られているように、通常の「ウインドウ・ベース」のグラフィカル・ユーザ・インターフェースと関連したディスプレイ出力は、ＧＵＩを構成するウインドウ、メニュー、タイトル・バー、境界線、スクロールバー、ビットマップ、アイコン、及び他の基本的な制御エレメントを無制限に含む多くのエレメントを有する。プラットフォーム独立のソフトウエア・プログラム（Ｊａｖａアプリケーション又はアプレットのような）の設計及び開発において、開発中の特定のソフトウエアが異種のコンピュータ・システム全体にわたって同じ「外観と操作性」を与えるかどうかを開発者（及びその他の人）が決定できることが望ましい。検証ツール２５は、プラットフォーム独立のアプリケーションが、ＩＢＭ ＯＳ／２プレゼンテーション・マネージャ、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ ＮＴ、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ ９５、Ｍａｃｉｎｔｏｓｈ、及びX Ｗｉｎｄｏｗｓ等のような既知のグラフィカル・ユーザ・インターフェースを介して、整合しているグラフィカル・ディスプレイ出力を与えるということを検証するための効率的な、しかも非常に正確な機構を提供する。

【００２３】図２は、異種のオペレーティング・システムを稼動させるコンピュータ全体にわたってソフトウエア・アプリケーション・グラフィカル・ディスプレイ出力を検証する場合に使用するための本発明の検証ツールの基本的オペレーションのフローチャートである。説明

の便宜上、以下の説明は、２つだけのオペレーティング・システムにわたってアプリケーションをテストすることに関連して行われる。しかし、複数のテスト・ランは任意の数の異種のコンピュータ・プラットフォームを含み得るので、これが本発明を制限するものではないことは当業者には明らかであろう。

【００２４】基本ルーチンは、コンピュータ又はテスト・サブシステム２６におけるハードウエアによって、又は検証ツール・ソフトウエアそのものにおいて実行可能である処理ステップを含む。そのルーチンは、ステップ３０において、すべてのテスト・ランが完了したかどうかを決定することによって始まる。前述のように、所与のテスト・ランは、そのテストによって評価されるオペレーティング・システム・プラットフォームのうちの所与の１つにおいて所与のアプリケーションに関して所定の処理ステップ（後述される）を伴う。ステップ３０におけるテストの結果が肯定的である場合、ルーチンは、詳しく後述されるステップ４０にブランチする。しかし、ステップ３０におけるテストの結果が否定的である場合、ルーチンはステップ３２において継続し、次のテストを行う。以下のステップは基本オペレーションを述べる。

【００２５】ステップ３４において、テスト中のアプリケーションが所与の実行ポイントに達したかどうかを決定するためのテストが行われる。本発明によれば、検証ツールによってテストされるアプリケーションは複数のテスト・ランを通して、しかし、所与の実行ポイントにおいて評価される。この方法では、開発者は、所与の実行ポイント全体にわたって変わらない動作特有のＧＵＩ構造を、その分析によって（望ましくは、後で明らかになるような統計的分析を使用することによって）取り除くことが可能である。所与の実行ポイントがユーザによってプリセットされるか、それが或るシステム制御の下に変化することがあることが望ましい。ステップ３４におけるテストの結果が否定的である場合、ルーチンは循環し、そのステップを反復し続ける。しかし、ステップ３４におけるテストの結果が肯定的である場合、それは所与の実行ポイントが特定のテスト・ランに達したことを表すので、ルーチンはステップ３４において継続する。

【００２６】この時点で、そのルーチンは、テスト中のアプリケーションによって発生される一対のグラフィカル出力フレームを処理する。望ましい実施例では、その一対のフレームは、テスト・サブシステムにおけるビデオ・フレーム・グラバを使用して捕捉され、そして、その対は、所与の実行ポイント（アプリケーションにおける）の直前にＧＵＩ上に表示される特定のフレーム、及び所与の実行ポイントの直後にＧＵＩ上に表示される特定のフレームを表す。フレームは、表示イメージのすべての或いは所与の部分を含む。その選択された特定のフレームは連続的なフレームである必要はなく、或いは、同じフレーム対の関係が複数のテスト・ラン全体にわたって使用される場合、前述の特定のタイミング関係を持つ必要もない。

【００２７】次に、ルーチンはステップ３８において継続し、グラフィカル出力フレームの対を処理する。望ましい実施例によれば、ステップ３８は、これらのフレームのビット単位の排他的ＯＲを行い、それによって、特定のテスト・ランに対するデルタ〝Δ〟を発生することを伴う。従って、デルタ値 Δ_1 はこの特定のテスト・ランに対して発生されるであろう（この場合、アプリケーションは第１の所与のオペレーティング・システムにおいて実行される）。ビット単位の排他的ＯＲ論理オペレーションがフレーム全体に対して行われることが望ましいけれども、これは本発明の必要条件ではない。従って、例えば、フレーム対の任意の所与の部分は、同じ所与の部分が他のテスト・ラン中に評価される場合、この方法で処理可能である。別の方法として、同様のオペレーションが他のテスト・ラン中に実施される場合、排他的ＯＲ以外の何らかの他の論理オペレーションが使用可能である。

【００２８】この方法におけるフレームの対の処理は大きな利点を与える。特に、フレームの対全体にわたって変わらないすべてのＧＵＩ表示構成は、特定の実行ポイントに存在するフレームの「差」だけを残して、本質的にマスクされ、又は、取り除かれる。従って、ウインドウ境界線、タイトル・バー、スクロールバー、アイコン等、及びオペレーティング・システム特有な他の情報を含むそのようなディスプレイ構成は、その検証プロセスにおける更なる考察から外される。

【００２９】ステップ３９においてデルタ値が保存された後、制御はステップ４０に戻る。このステップにおいて、デルタ値｛Δ_1、Δ_2、・・・｝が記憶装置から検索される。これらの値の各々は、連続するフレーム対の出力（即ち、フレーム〝ｎ－１〟からフレーム〝ｎ〟まで）がアプリケーションにおける同じ所与の実行ポイントにおいて変わる程度を表す（そのアプリケーションが複数のオペレーティング・システム・プラットフォームの特定の１つにおいて実行される時）。ステップ４２において、ルーチンは、それらのデルタ値を分析するための所与の統計的方法を選択する。本発明によれば、複数のテスト・ラン全体にわたって発生されたデルタ値を比較するために、任意の適当な統計的方法が使用可能である。既知の統計的方法は、パターン・マッチング・アルゴリズム、ファジー・ロジック・アルゴリズム、一組の学習ルールを有する適応性推論エンジン、及び他の既知の又は今後開発される統計的評価ルーチンを無制限に含む。その特定のルーチンは、コンピュータにおいて稼動するソフトウエアにおいて実現される。望ましい場合には、そのステップは、任意の既知の又は今後開発される

決定的な分析方法を包含し得る。

【００３０】ステップ４４において、統計的方法がそのセットのデルタ値に適用される。次に、ステップ４４において、デルタ値が所与のユーザ選択可能な信頼性レベル（例えば、９５％）内に「一致」するかどうかを決定するために、テストが実行される。勿論、任意のタイプの統計的メトリックが使用可能である。ステップ４６におけるテストの結果が否定的である場合、ステップ４８において、アプリケーションのグラフィカル・ディスプレイ出力がテストされた複数のオペレーティング・システム・プラットフォーム全体にわたって整合しないという表示がユーザに与えられる。望ましい場合には、ステップ５０において、特定のディスプレイ出力変動を分離し及び識別するための更なるステップが実行可能である。しかし、ステップ４６におけるテストの結果が肯定的である場合、ルーチンはステップ５２に継続し、グラフィカル・ディスプレイ出力がテストされた複数のオペレーティング・システム全体にわたって「有効」であるという表示を発生させる。これは処理を終了させる。

【００３１】従って、図３のブロック図に示されるように、各テスト・ラン５４はデルタ値Δ_nを発生する。そこで、種々のデルタ値が統計的分析器５６に供給され、「一致」又は「拒否」を発生させる。もう１つの別の方法の出力は「中間」であり、従って、１つ又は複数のテスト・ランが繰り返される必要があるかもしれない。そのツールは、どの特定のグラフィカル・ディスプレイ出力情報がその拒絶に貢献したか又は生じさせたかを識別する特定の情報を評価及び出力するための診断ルーチンも含み得る。そこで、この情報は、その問題を修正する（例えば、アプリケーション・コードを再処理することによって）ように開発者によって使用可能である。

【００３２】従って、本発明は、テスト中のアプリケーションが種々のオペレーティング・システム・プラットフォーム全体にわたって実行される時、連続的なグラフィカル・ディスプレイ出力フレームを都合よく捕捉する。そこで、それらのフレームは、所与の実行ポイントにおいて時不変性のすべてのオペレーティング・システム特有のＧＵＩ及びプログラム発生された構成をマスク・アウトするように処理される。所与の統計的分析を使用して、各テスト・ランに対して発生されたデルタ値を比較することによって、検証ツールがプログラム出力の整合性を確認する。これは、特にプラットフォーム独立のアプリケーションに関して、プログラム開発の時間、努力、及び経費をかなり減少させる。そのツールを使用してテスト及びデバッグ可能な特定のアプリケーションは、Ｊａｖａアプリケーション又はＪａｖａアプレットである。Ｊａｖａアプレットの場合、整合性を更に検証するためにアプレット・ビューアを使用することが可能である。

【００３３】図４は、所与のアプリケーションが連続的なテスト・ランにおいて同種のプラットフォームに関してテストされるという本発明の修正バージョンのフローチャートを示す。この実施例は、同じ又は実質的に同じ実行環境を使用する複数の実行全体にわたってアプリケーションが整合性を持って動作することをアプリケーション開発者が確認することを可能にする。従って、図４の実施例では、オペレーティング・システム・プラットフォームは同じであると思われる。

【００３４】前の実施例の場合のように、基本ルーチンは、コンピュータ又はテスト・サブシステム２６におけるハードウエアによって実行される処理ステップ又は検証ツール・ソフトウエアそのものにおける処理ステップを含む。そのルーチンはステップ６０において始まり、すべてのテスト・ランが完了したかどうかを決定する。この実施例では、所与のテスト・ランは、同じオペレーティング・システム・プラットフォーム（しかし、恐らく、異なるハードウエア構成を持った）において実行される所与のアプリケーションに関して或る処理ステップ（後述される）を伴う。ステップ６０におけるテストの結果が肯定的である場合、ルーチンはステップ７０にブランチする。ステップ７０に関しては、以下で詳しく述べることにする。しかし、ステップ６０におけるテストの結果が否定的である場合、ルーチンは、次のテストを行うためにステップ６２において継続する。

【００３５】ステップ６４において、テスト中のアプリケーションが実行ポイントに達したかどうかを決定するためのテストが行われる。ステップ６４におけるテストの結果が否定的である場合、ルーチンは循環し、そのステップを反復し続ける。しかし、ステップ６４におけるテストの結果が肯定的である場合、それは、特定のテスト・ランに対する所与の実行ポイントが到達したことを表し、ルーチンはステップ６６において継続する。

【００３６】図３のルーチンに関して前述したように、そのルーチンはステップ６８に継続し、ビデオ・フレーム・グラバによって捕捉されたグラフィカル出力フレームの対を処理する。望ましい実施例によれば、ステップ６８が再び関与して、フレームのビット単位排他的ＯＲを取り、それによって、特定のテスト・ランに対するデルタ〝Δ〟値を発生する。従って、デルタ値〝Δ_1〟がこの特定なテスト・ランに対して再び発生されるであろう。そこで、ステップ６９において、そのデルタ値は保存され、制御はステップ７０に戻る。このステップにおいて、デルタ値｛Δ_1, Δ_2, ・・・｝が記憶装置から検索される。ステップ７２において、ルーチンは、デルタ値を分析するための所与の統計的方法を選択する。

【００３７】ステップ７４において、統計的方法がそのセットのデルタ値に適用される。そこで、テストが行われ、デルタ値が所与のユーザ選択可能な信頼性レベル内の値に「一致」するかどうかを決定する。ステップ７６におけるテストの結果が否定的である場合、ステップ７

８において、アプリケーションのグラフィカル・ディスプレイ出力がテストされつつあるシステム・プラットフォーム全体にわたって整合していないという表示がユーザに与えられる。望ましい場合には、ステップ８０において更なるステップが実行され、特定のディスプレイ出力変動を識別するようにしてもよい。しかし、ステップ７６におけるテストの結果が肯定的である場合、ルーチンはステップ８２において継続し、グラフィカル・ディスプレイ出力がテストされる複数のオペレーティング・システム全体にわたって「有効」であるという表示を発生する。テストの結果が不確定であるか或いは曖昧である場合、ルーチンは適切な表示を与えることも可能である。これは処理を完了させる。

【００３８】本発明がその概念の範囲内で変更又は修正可能であり、その範囲内のままであってもよいことは当業者には明らかであろう。従って、例えば、検証ツールに「差」又はデルタ値を発生される代わりに、個々のフレーム対（所与のテスト・ランにおける）が「同じ」である範囲を決定することが望ましいこともある。従って、「差分表現（ｄｉｆｆｅｒｅｎｃｉｎｇ）」技法が望ましいけれども、それは本発明を限定するものではない。更に、デルタ値又は他の値に関する統計的分析がテスト結果を検証する望ましい方法であるけれども、決定論的な分析技法が使用可能であることも当業者には明らかであろう。そのような変更はすべて本発明の技術的範囲内である。

【００３９】本発明の望ましい実現方法の１つはアプリケーション、即ち、例えば、コンピュータのランダム・アクセス・メモリに常駐し得るコード・モジュールにおける命令（プログラム・コード）のセットである。その命令のセットは、コンピュータによって要求されるまで、他のコンピュータ・メモリ、例えば、ハードディスク・ドライブ、或いは光ディスク（ＣＤ－ＲＯＭにおいて使用するための）又はフロッピ・ディスク（フロッピ・ディスク・ドライブにおいて使用するための）のような取り外し可能なメモリに記憶可能であり、或いはインターネット又は他のコンピュータ・ネットワークを介してダウンロード可能である。従って、本発明は、コンピュータにおいて使用するためのコンピュータ・プログラム製品として実現可能である。

【００４０】更に、前述の種々の方法はソフトウエアによって活性化され又は再構成された汎用のコンピュータにおいて都合よく実施可能であるけれども、そのような方法が、ハードウエア、ファームウエア、又は必要な方法ステップを遂行するように構成された更に特殊な装置において実行可能であることも当業者には明らかであろう。

【００４１】必要なことではないけれども、本発明を構成する種々なプロセスは同じホスト・マシン上に、又はネットワーク（例えば、インターネット、イントラネット、広域ネットワーク（ＷＡＮ）、又はローカル・エリア・ネットワーク（ＬＡＮ））を介して相互接続された相異なるマシン上に常駐可能である。従って、本発明を実行する機械は、他のコンピュータへの接続を確立するために適正なネットワーキング・ハードウエアを有する。例えば、その機械は、トークン・リング又はイーサネット・アダプタを介して実行するネットワークへのＴＣＰ／ＩＰ又はＮＥＴＢＩＯＳ接続を持つものでもよい。

【００４２】上記の特定の実施例が、本発明の同じ目的を実行するための別の手法を修正又は設計するための基本として容易に利用可能であることは当業者には明らかであろう。又、そのような等価な構成が、特許請求の範囲に記載されたような本発明の趣旨及び技術的範囲から逸脱しないことも当業者には明らかであろう。

【００４３】まとめとして、本発明の構成に関して以下の事項を開示する。

【００４４】（１）所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する方法にして、前記それぞれのオペレーティング・システムの各々に基づいて前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて、一対のグラフィカル・ディスプレイ出力フレームを捕捉するステップと、前記それぞれのオペレーティング・システムの各々に対して、前記一対のフレームにわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために前記グラフィカル・ディスプレイ出力フレームの各対を処理するステップと、所与の前記ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって所与の基準を満たすことを検証するために、それぞれのオペレーティング・システムの各々に対して発生された所与の情報を比較するステップと、を含む方法。

（２）所与の前記ソフトウエア・アプリケーションはＪａｖａで書かれたアプリケーションであることを特徴とする上記（１）に記載の方法。

（３）前記一対のフレームは連続的なグラフィカル・ディスプレイ出力フレームであることを特徴とする上記（１）に記載の方法。

（４）前記比較するステップは統計的アルゴリズムの実行を含むことを特徴とする上記（１）に記載の方法。

（５）前記所与の基準は前記所与の情報が所与の信頼性レベル内の情報と等しいかどうかをテストすることを特徴とする上記（１）に記載の方法。

（６）前記所与の信頼性レベルはユーザ選択可能であることを特徴とする上記（５）に記載の方法。

（７）前記処理するステップは前記一対のフレームを含

むフレームにおける排他的ＯＲ論理オペレーションの遂行を含むことを特徴とする上記（１）に記載の方法。

（８）前記排他的ＯＲ論理オペレーションはビット単位の基準で遂行されることを特徴とする上記（７）に記載の方法。

（９）Ｊａｖａアプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する方法にして、前記それぞれのオペレーティング・システムの各々に基づいてＪａｖａアプリケーションにおける所与の実行ポイントにおいて、連続するグラフィカル・ディスプレイ出力フレームを捕捉するステップと、前記それぞれのオペレーティング・システムの各々に対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクするデルタ値を発生するために、前記連続するグラフィカル・ディスプレイ出力フレームを処理するステップと、前記Ｊａｖａアプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するために、前記それぞれのオペレーティング・システムの各々に対して発生された前記デルタ値を統計的に比較するステップと、を含む方法。

（１０）前記Ｊａｖａアプリケーションはアプレットであることを特徴とする上記（９）に記載の方法。

（１１）前記処理するステップはビット単位の排他的ＯＲ論理オペレーションであることを特徴とする上記（９）に記載の方法。

（１２）前記比較するステップは前記デルタ値に関して統計的分析を行うことを含むことを特徴とする上記（９）に記載の方法。

（１３）前記統計的分析はパターン・マッチング・アルゴリズムであることを特徴とする上記（１２）に記載の方法。

（１４）前記統計的分析はファジー・ロジック・アルゴリズムであることを特徴とする上記（１２）に記載の方法。

（１５）前記統計的分析は神経ネットワーク・アルゴリズムであることを特徴とする上記（１２）に記載の方法。

（１６）所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システム全体にわたって整合していることを検証する場合に使用するためのコンピュータ読み取り可能な媒体におけるコンピュータ・プログラム製品にして、前記それぞれのオペレーティング・システムの各々に基づいて前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて動作し、グラフィカル・ディスプレイ出力フレームの対を捕捉するための手段と、前記それぞれのオペレーティング・システムに対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために、前記捕捉手段に応答して前記グラフィカル・ディスプレイ出力フレームの各対を処理するための手段と、前記所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するために、前記処理するための手段に応答して、前記それぞれのオペレーティング・システムの各々に対して発生された所与の情報を比較するための手段と、を含むコンピュータ・プログラム製品。

（１７）前記所与のソフトウエア・アプリケーションはＪａｖａで書かれたアプリケーションであることを特徴とする上記（１６）に記載のコンピュータ・プログラム製品。

（１８）前記捕捉するために手段はビデオ・フレーム・グラバであることを特徴とする上記（１６）に記載のコンピュータ・プログラム製品。

（１９）前記処理するための手段は排他的ＯＲ論理オペレーションであることを特徴とする上記（１６）に記載のコンピュータ・プログラム製品。

（２０）前記比較するための手段は統計的分析であることを特徴とする上記（１７）に記載のコンピュータ・プログラム製品。

（２１）前記統計的分析はパターン・マッチング・アルゴリズムであることを特徴とする上記（２０）に記載のコンピュータ・プログラム製品。

（２２）前記統計的分析はファジー・ロジック・アルゴリズムであることを特徴とする上記（２０）に記載のコンピュータ・プログラム製品。

（２３）前記統計的分析は神経ネットワーク・アルゴリズムであることを特徴とする上記（２０）に記載のコンピュータ・プログラム製品。

（２４）プロセッサと、所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力がそれぞれのオペレーティング・システムに全体にわたって整合していることを検証する場合に使用するためのツールと、を含み、前記ツールは、前記それぞれのオペレーティング・システムの各々に基づいて前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて動作し、一対のグラフィカル・ディスプレイ出力フレームを捕捉するための手段と、前記それぞれのオペレーティング・システムに対して、前記一対のフレーム全体にわたって時不変性である前記それぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために、前記捕捉手段に応答して前記グラフィカル・ディスプレイ出力フ

レームの各対を処理するための手段と、前記所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するように、前記処理するための手段に応答して、前記それぞれのオペレーティング・システムの各々に対して発生された所与の情報を比較するための手段と、を含むコンピュータ。

（２５）前記所与のソフトウエア・アプリケーションはＪａｖａで書かれたアプリケーションであることを特徴とする上記（２４）に記載のコンピュータ。

（２６）前記捕捉するために手段はビデオ・フレーム・グラバであることを特徴とする上記（２４）に記載のコンピュータ。

（２７）前記処理するための手段は排他的ＯＲ論理オペレーションであることを特徴とする上記（２４）に記載のコンピュータ。

（２８）前記比較するための手段は統計的分析であることを特徴とする上記（２４）に記載のコンピュータ。

（２９）オブジェクト指向プラットフォーム独立のプログラミング環境において最初に書かれたソフトウエア・アプリケーションが第１及び第２オペレーティング・システムのそれぞれにおいて稼動するそれぞれのウインドウ環境において実行される時、前記ソフトウエア・アプリケーションが整合しているグラフィカル出力を有することを検証する方法にして、前記それぞれのオペレーティング・システムの各々に基づいて前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて、一対のグラフィカル・ディスプレイ出力フレームを捕捉するステップと、前記それぞれのオペレーティング・システムに対して、前記一対のグラフィカル・ディスプレイ出力フレーム全体にわたって時不変性であるそれぞれのオペレーティング・システムに対するグラフィカル・ディスプレイ出力の構成をマスクする所与の情報を発生するために前記グラフィカル・ディスプレイ出力フレームの各対を処理するステップと、前記所与のソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記それぞれのオペレーティング・システム全体にわたって整合していることを検証するために、前記それぞれのオペレーティング・システムの各々に対して発生された所与の情報を統計的に比較するステップと、を含む方法。

（３０）前記オブジェクト指向プラットフォーム独立のプログラミング環境はＪａｖａであることを特徴とする上記（２９）に記載の方法。

（３１）前記ソフトウエア・アプリケーションがそれぞれのテスト・システム・コンフィギュレーション全体にわたって実行される時、前記ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が整合していることを検証する方法にして、前記ソフトウエア・アプリケーションがそれぞれのテスト・システム構成の各々に基づいて実行される時、前記ソフトウエア・アプリケーションにおける所与の実行ポイントにおいて、連続するグラフィカル・ディスプレイ出力フレームを捕捉するステップと、それぞれの前記テスト・システム・コンフィギュレーションに対して、前記グラフィカル・ディスプレイ出力フレームの対にわたって時不変性であるグラフィカル・ディスプレイ出力の構成をマスクするデルタ値を発生するために、前記連続するグラフィカル・ディスプレイ出力フレームを処理するステップと、前記ソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力が前記テスト・システム・コンフィギュレーション全体にわたって整合していることを検証するために、前記テスト・システム・コンフィギュレーションの各々に対して発生された前記デルタ値を統計的に比較するステップと、を含む方法。

（３２）前記テスト・システム・コンフィギュレーションの少なくとも２つは所与のオペレーティング・システムを利用することを特徴とする上記（３１）に記載の方法。

（３３）前記テスト・システム・コンフィギュレーションの少なくとも２つは相異なるオペレーティング・システムを利用することを特徴とする上記（３１）に記載の方法。

【図面の簡単な説明】

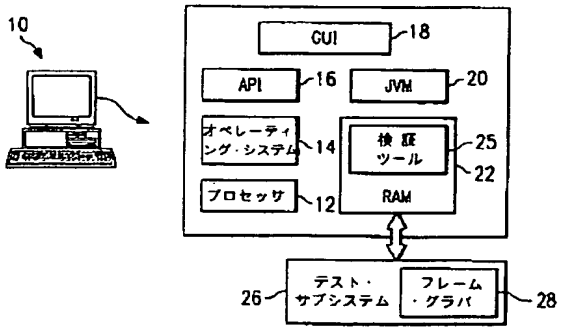
【図１】本発明が全体的に又は部分的に実施される代表的なコンピュータのブロック図である。

【図２】異種のオペレーティング・システムを稼動させるコンピュータ全体にわたってソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力を検証する場合に使用するための本発明の検証ツール基本オペレーションのフローチャートである。

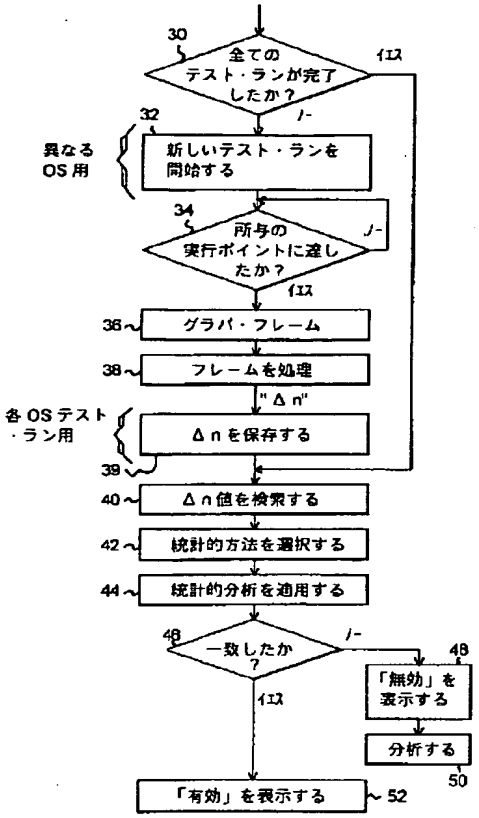
【図３】本発明の検証ツールの主要な処理機能を示すブロック図である。図２のツールの望ましい動作方法を示すフローチャートである。

【図４】同種のオペレーティング・システムを稼動させるコンピュータ全体の複数のテスト・ラン全体にわたってソフトウエア・アプリケーションのグラフィカル・ディスプレイ出力を検証する場合に使用するための本発明の検証ツール基本オペレーションのフローチャートである。

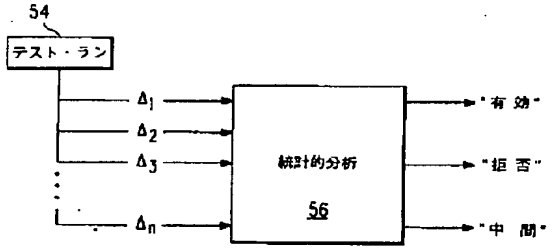
【図１】


10
CUI
18
API
16
JVM
20
オペレーティング・システム
14
検証ツール
25
22
プロセッサ
12
RAM
テスト・サブシステム
26
フレーム・グラバ
28

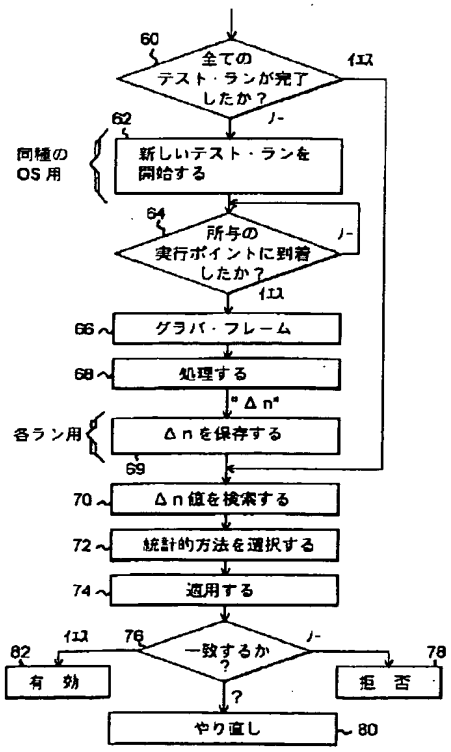
【図２】


30
全てのテスト・ランが完了したか？
イエス
ノー
32
異なるOS用
新しいテスト・ランを開始する
34
所与の実行ポイントに達したか？
ノー
イエス
36
グラバ・フレーム
38
フレームを処理
"Δn"
各OSテスト・ラン用
Δnを保存する
39
40
Δn値を検索する
42
統計的方法を選択する
44
統計的分析を適用する
46
一致したか？
ノー
48
「無効」を表示する
分析する
50
イエス
「有効」を表示する
52

【図３】


54
テスト・ラン
Δ1
Δ2
Δ3
Δn
統計的分析
56
"有効"
"拒否"
"中間"

【図４】


60
全てのテスト・ランが完了したか？
イエス
ノー
62
同種のOS用
新しいテスト・ランを開始する
64
所与の実行ポイントに到着したか？
ノー
イエス
66
グラバ・フレーム
68
処理する
"Δn"
各ラン用
Δnを保存する
69
70
Δn値を検索する
72
統計的方法を選択する
74
適用する
76
一致するか？
イエス
ノー
82
有効
78
拒否
？
やり直し
80